Claims:

Please amend claims 1 and 5-7. Please add new claims 11-19. Claims 2 and 9 were previously cancelled. No new matter has been added.

Listing of Claims:

1. (Currently Amended) A method comprising:

executing corresponding instruction threads in parallel as a leading thread and a trailing thread;

speculatively saving a result from a first instruction executed in the leading thread and speculatively saving a result from a second instruction corresponding to the first instruction executed in the trailing thread to a memory having extensions for speculative storage;

comparing the results saved in the memory;

committing a single set of instructions based on the compared result; and

deferring ~~external~~ updates **from one or more external agents to one or more memory locations accessed by the instruction threads** until **the single set of instructions has been committed** ~~completion of the step of committing~~.

2. (Cancelled)

3. (Previously Presented) The method of claim 1, wherein the corresponding instruction threads are epoch instruction threads.

4. (Previously Presented) The method of claim 3, wherein a location read by the leading thread during an epoch contains same value as that read by the leading thread when the corresponding read by the trailing thread load occurs.

5. (Currently Amended) An apparatus comprising:

a means for executing parallel threads as a leading thread and a trailing thread;

a means for speculatively saving the results from the executed threads in a memory having extensions for speculative storage;

a means for comparing the results saved in the memory;

a means for committing a single set of **instructions** ~~**thread**~~ based on the compared result; and

a means for deferring **one or more** ~~**external**~~ updates **from one or more external agents to one or more memory locations accessed by the parallel threads** until **the single set of instructions has been committed** ~~**completion of the step of committing**~~.

6. (Currently Amended) The apparatus of claim 5 wherein the **parallel** ~~**executed**~~ threads **each execute an instance of an epoch** ~~**are epoch threads**~~.

7. (Currently Amended) The apparatus of claim **5** ~~**6**~~, wherein each **of the one or more updates is a write from one of the one or more external agents to at least**

one of the one or more memory locations accessed by the parallel threads ~~**epoch is executed twice**~~.

8. (Original) The apparatus of claim 5 wherein a location having a first value when loaded by the leading thread during an epoch contains the first value with the corresponding load by trailing thread loads occurs.

9. (Cancelled)

10. (Previously Presented) The apparatus of claim 8 wherein the single set is committed if the compare result matches.

11. (New) An apparatus comprising:

a thread execution logic to:

execute parallel threads as a leading thread and a trailing thread;

speculatively save the results from the executed threads in a memory having extensions for speculative storage;

compare the results saved in the memory; and

commit a single set of instructions based on the compared result; and

a conflicting access detection mechanism to:

defer one or more updates from one or more external agents to one or more memory locations accessed by the parallel threads until the single set of instructions has been committed.

12. (New) The apparatus of claim 11 wherein the parallel threads each execute an instance of an epoch.

13. (New) The apparatus of claim 12, wherein each of the one or more updates is a write from one of the one or more external agents to at least one of the one or more memory locations accessed by the parallel threads.

14. (New) The apparatus of claim 13, wherein each of the one or more writes are flagged for deferral when the write occurs to a location within a cache block that has been accessed by the parallel threads.

15. (New) The apparatus of claim 14, wherein the conflicting access detection mechanism further is operable to:

not defer a first write from the one or more writes when the first write targets a different location within the cache block than any locations accessed by the parallel threads.

16. (New) The apparatus of claim 14, wherein the conflicting access detection mechanism is further operable to:

not defer a first write from the one or more writes when the first write is determined to be targeting a memory location that has completed accesses from

both parallel threads and a same value has been retrieved from both parallel thread accesses.

17. (New) The apparatus of claim 14, wherein the conflicting access detection mechanism is further operable to:

defer a first write from the one or more writes when the first write is to a memory location that one of the parallel threads has completed an access to and the other of the parallel threads has not completed an access to.

18. (New) The apparatus of claim 11 wherein a location having a first value when loaded by the leading thread during an epoch contains the first value with the corresponding load by trailing thread loads occurs.

19. (New) The apparatus of claim 18 wherein the single set is committed if the compare result matches.